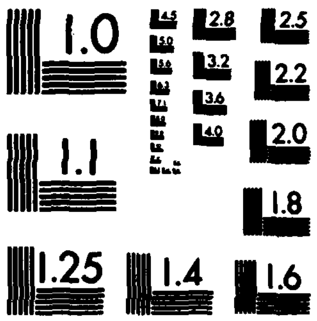
1.0
4.5
5.0
5.6
6.3
7.1
8.0
9.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

This final report was submitted by the Aerospace Corporation, El Segundo, CA 90245 under Contract No. F04701-82-C-0083 with the Space Division, Deputy for Logistics and Acquisitions, P.O. Box 92960, Worldway Postal Center, Los Angeles, CA 90009. It was reviewed and approved for The Aerospace Corporation by J. R. Coge, Electronics and Optics Division, Engineering Group. Al Carlan was the project engineer.

This report has been reviewed by the Office of Information and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

FOR THE COMMANDER

APPROVED

[illegible]

STEPHEN A. HUNTER, LT COL, USAF
Director, Speciality Engineering
and Test

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
SD-TR-83-20

2. GOVT ACCESSION NO.
AD-A128 019

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
State-of-the-Art Assessment of Testing and Testability of Custom LSI/VLSI Circuits
Vol VIII: Fault Simulation

5. TYPE OF REPORT & PERIOD COVERED
Interim

6. PERFORMING ORG. REPORT NUMBER
TR-0083(3902-04)-1

7. AUTHOR(s)
M.A. Breuer & Associates
and
A.J. Carlan, Aerospace Technical Director

8. CONTRACT OR GRANT NUMBER(s)
F04701-80-C-0081
F04701-81-C-0082
F04701-82-C-0082

9. PERFORMING ORGANIZATION NAME AND ADDRESS
M.A. Breuer & Associates
16857 Bosque Dr.
Encino, CA 91436

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Space Division
Air Force Systems Command
Los Angeles, Calif. 90009

12. REPORT DATE
October 1982

13. NUMBER OF PAGES
28

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
The Aerospace Corporation
El Segundo, Calif. 90245

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Parallel simulation
Deductive simulation
Concurrent simulation
Gate-level modeling
Functional-level modeling
Digital networks simulation
Fault simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Fault simulation is widely used by industry in such applications as scoring the fault coverage of test sequences and construction of fault dictionaries. For use in testing VSLI circuits a simulator is evaluated by its accuracy i.e. modelling capability. To be accurate simulators must employ multi-valued logic in order to represent unknown signal values, impedance, signal transitions etc, circuit delays such as transport rise/fall, inertial, and the fault modes it is capable of handling. Of the three basic fault simulators now in use (parallel, deductive and concurrent) concurrent fault simulation appears most promising.

DD FORM 1473 (FACSIMILE)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Fault simulation is widely used in industry, primarily in such applications as scoring the fault coverage of a test sequence, construction of fault dictionaries, and as an aid in test pattern generation by using it to determine the set of faults detected by a candidate test.

There are three basic fault simulation approaches in current use, known as parallel, deductive and concurrent simulation. Of these, concurrent fault simulation (which is the newest of the three) appears to be the most promising for VLSI. It appears to be the fastest and the most accurate in terms of timing, and is very compatible with functional models, multi-valued logic and adaptability to new fault modes and new logic primitives. Its major disadvantage is in the very large amount of memory that it requires.

An important issue in the evaluation of a simulator is its accuracy, that is, its modeling capability. Modeling accuracy is determined by the types of primitives available to the simulator. Gate-level modeling of a network results in high state and timing accuracy but may not be practical for VLSI. Functional-level modeling is less accurate but the modeling effort is reduced and the simulation speed is improved, thus making it more suitable for simulation of VLSI circuits. Development of efficient mixed-level simulators which can handle circuit descriptions at both a low-level and a high-level is a desired

objective. On the issue of logic values, the use of two-valued logic (0,1) is totally inadequate. Simulators must employ multi-valued logic in order to represent unknown signal values, high-impedance, various signal transitions, pulses, etc. Accurate simulation must also take into consideration circuit delays, such as transport, ambiguity, rise/fall and inertial delays. Modeling of delays in complex functional primitives is particularly difficult.

Another important feature of a fault simulator is the fault modes that it is capable of handling. Though most simulators primarily handle stuck-at faults, this may not be adequate for some VLSI technologies. Nor is it adequate for high-level modeling.

Most fault simulators in current use handle stuck-at faults and employ parallel fault simulation at the gate level with three or four logic values. Most of the new simulators or the ones under development employ concurrent fault simulation with some functional or mixed-level modeling capability under development. Additional trends in fault simulation include the processing of more complex fault modes, allowing for dynamic switching of models, and the development of new techniques to improve simulation efficiency and accuracy.

CHAPTER 1 INTRODUCTION

Digital networks simulation is the process of building and exercising a model of a circuit on a digital computer, i.e., the evaluation for some input sequence of signal values in the modeled circuit as a function of time.

Simulation of digital networks is divided into two major types: good network (or true-value) simulation used in logic verification, and fault simulation where the behavior of the digital network is simulated in the presence of faults. Fault simulation involves the insertion of one or more faults into the computer model of the simulated network. It replaces the physical insertion of faults into the constructed network. It enables the evaluation of a network's behavior under faulty conditions without the actual physical construction of the network, or for such faults that cannot be inserted into the physical network due to the lack of external access to some internal signals. The good network is going to be called the *good machine,* while the same network with a fault inserted is going to be called a *faulty machine.* One simulation of a network under either no fault, single fault, multiple faults, or multiple faulty machine conditions is called a *simulation pass.*

Fault simulation is usually done at the gate level where the network primitives are gates (AND, OR, etc.). Newer simulators incorporate higher level primitives such as flip-flops, arithmetic units, multiplexers, registers, etc.

The major uses of fault simulation are as follows:

1. Construction of fault dictionaries, which indicate the input patterns under which fault symptoms appear on network outputs.
2. Obtaining test coverage, that is, the scoring of a given test sequence in terms of the percentage and the identity of detected and undetected faults.
3. During test pattern generation (TPG), in determining the set of faults which are detected by a candidate test. This is used to support an algorithmic, heuristic or random automatic TPG system by providing precise verification of the results of test patterns.
4. Support the interactive development of test and diagnostic programs. That is done in steps by using intermediate results produced by the simulator.
5. Analyze the network operation under fault conditions.

It should be noted that fault simulation is not needed for all types of testing or for all types of hardware, e.g., RAM, ROM and PLA.

CHAPTER 2 DEFINITIONS

a) Techniques

"Event-directed" (also known as "selective trace") simulation evaluates the state of only those elements for which one or more inputs have changed. If the output of an element changes, the elements fed by it are scheduled to be simulated (evaluated). This is a very important technique for speeding up simulation.

The description of the circuit to be simulated can be utilized in one of two general ways: in a "compiler-driven" simulator, the circuit description is treated as a source program in a high-level programming language. Simulation of the circuit consists of executing the program compiled by a preprocessor. A major disadvantage of of a compiler-driven simulator is that selective trace cannot be used.

A "table-driven" simulator uses tables to describe the circuit. The simulation program uses these tables as data. Table driven simulators allow for the use of selective trace.

Practically all gate-level simulators that have been recently built are table-driven, event-directed.

b) Fault Modes

A physical fault in a digital network may be classified as logical or parametric. A logic fault changes the logic function of one or more circuit elements or their inputs. Parametric faults frequently affect a circuit parameter which may affect circuit speed, current or voltage, and they cannot be treated as logic faults.

Physical faults are processed by a simulator via fault models. The stuck-at fault model (see report AC 2.81) is the model most widely used in fault simulation, especially when the circuit is modeled at the gate level. A stuck-at-one (s-a-1) fault assumes that a signal in the network becomes fixed at the constant value of logical one. Similarly for s-a-0 fault and s-a-Z faults (Z represents high impedance). Other popular gate level fault modes are bridge faults, i.e., the shortening of two leads and various open circuit faults and shorts (e.g., shorted diode).

At the functional level, fault modes that are being used include stuck-at signals, bridges, and in advanced simulators faults which change a function f into an arbitrarily different function g.

c) Miscellaneous

A "hazard" is a transient error or spike caused by circuit delays. Hazards may cause an asynchronous circuit to enter an erroneous stable state.

In asynchronous circuits signals which are intended to change concurrently cannot be assumed to change in synchronism due to the absence of a synchronizing clock. The behavior of the circuit may depend on the actual order in which the signals change, resulting in a "race" conditions.

CHAPTER 3 MODELING AND PROCESSING OF MODELS

The inputs to a digital fault simulator usually consist of the following:

1. Description of the circuit to be simulated.
2. Input data to be simulated.
3. Initial value of memory states.
4. Faults to be simulated.
5. Signals to be monitored.

The circuit description consists of the topology of the circuit and the circuit element types, along with a list of primary inputs and primary outputs including test points. Delay parameters and circuit restrictions such as fanout restrictions may also be included. Such inputs are usually described in a high-level language.

Modeling is the process of describing a circuit in terms of the primitives of the processing system. The objective is to model the circuit such that the results from the simulator will correspond to the signal values in the actual circuit.

The accuracy of a fault simulator is determined primarily by its modeling capability, the number of logic values used, and how time and circuit delay is processed.

a) Modeling

The accuracy of modeling a digital system is determined by the types of primitives that the simulator has available in describing the system, i.e., pass transistors, gates (AND, OR), flip flops, functions, structures (PLAs, RAMs), RT (behavioral, such as Boolean equations descriptions).

Most existing simulators employ a gate-level model, where the network primitives are gates. Such simulation requires detailed gate-level descriptions of networks, i.e., detailed interconnections between gates and delay specifications. This results in accurate state and timing information. In functional level modeling the primitives are devices such as registers, arithmetic units, memories, etc. [Chappell 1976, Abramovici 1977, Malek 1978, Schuler 1979, d'Abreu 1980, Ulrich 1980]. This form of modeling lacks some of the details, in terms of state and timing, that the gate-level models have. Consequently, the resulting accuracy from functional-level simulation is lower than that of gate-level simulation. At the same time, the modeling effort is reduced and the simulation speed is improved.

The issues involved in the design of a functional level concurrent fault simulator are discussed in [Abramovici 1977] while [Schuler 1977] describes improved techniques for concurrent fault simulation including the modeling and simulation of ROMs and RAMs at the RTL level. The application of functional simulation to fault diagnosis is considered in [Malek 1978].

Mixed-level simulators allow the mixing of different levels of modeling, such as gate level and functional level modeling [Strange 1977].

Northcutt describes the design and implementation of a high level fault insertion mechanism for the true-value ISPS (Instruction Set Processor Specification) simulator [Northcutt 1980]. The ISPS is a mature, interactive high level simulator. The faults which can be simulated include: permanent and transient; deterministic and probabilistic; stuck-at, bridge, data, control and operation. This new simulator is currently under evaluation. One project uses both an ISPS simulation and a gate-level simulation of the same machine to determine the degree to which the two can provide comparable coverage. Their relative speeds and efficiency will also be compared. The ISPS simulation has been used effectively to validate the proper operation of hardware error detection and correction networks. Other uses are contemplated for this simulator, but the verdict is not yet in as to its usefulness.

b) Logic Values

At least 4 logic values are necessary to accurately process logic namely 0, 1, u (unknown) and Z (high impendance). It is sometimes useful to have additional logic levels available to describe transitions ($0\rightarrow 1$ and $1\rightarrow 0$), hazards, weak 0 and weak 1 (for MOS logic).

c) Timing Accuracy

A simulator calculates the logic value of a signal line as a function of time. Hence accurate simulation must take into consideration circuit delays. Most component models and simulation techniques can be categorized as being either pessimistic or optimistic models of the real circuit. Usually a more detailed timing model is used for design verification (true-value simulator) than is used for fault simulation. The problem of modeling delays in complex functional primitives is particularly difficult.

Circuit delays can be characterized by one or more of the following delay models.

1. Pure (transport) delay — The delay introduced into a signal propagating through an element or through a wire. Simulators using "assignable delays" assign to each element its pure delay which is typically a multiple of some common unit. On the other hand "unit delay" simulators assign the same delay to all elements.

2. Ambiguity delay — For most elements the exact pure delay is not known. It is possible to model the delay of such an element by a pair of delay values giving the minimum Δ_m and the maximum Δ_M delay through the element. These delays define an ambiguity region of duration $(\Delta_M - \Delta_m)$. More than two logic values are necessary in order to model ambiguity delays in order to be able to represent signals in the ambiguity region.

3. Rise/Fall delay — For some elements (e.g., MOS devices) the output response rise and fall time are different. Such elements can be modeled by assigning two delays Δ_R and Δ_F to the output of the element.

4. Inertial delay — The minimum duration Δ_I, for which an input change must persist in order for the element to switch states.

Pure delay (sometimes called nominal delay) is the easiest type to simulate. Simulation of rise/fall delays is somewhat more complex, while simulation of ambiguity delay is still more difficult. Inertial delay is the most complex to simulate.

CHAPTER 4 FAULT SIMULATION TECHNIQUES

The following are the major fault simulation techniques [Breuer 1976]:

1. single fault simulation
2. parallel fault simulation
3. deductive fault simulation
4. concurrent fault simulation.

In single fault simultion only one faulty machine is simulated in one simulation pass. A network for which k faults need to be simulated requires (k+1) simulation passes (one simulation pass for the good machine).

Parallel fault simulation [Szygenda 1972b, Thompson 1975b] has been the most widely used technique in industry. N faulty machines are simulated in one simulation pass, where N is related to the host computer word length. For example (for two-valued logic), when the host computer word length is N+1, a bit position in a computer word represents the signal value on a node of the simulated network for one of the N faulty machines or for the good machine. The number of simulation passes is reduced to k/N. Since each bit of a computer word must be processed independently of the other bits in that word, element evaluation routines normally consist of logical operations. If the simulator is table-driven event-directed, the values of all the signals on signal line a must be recomputed whenever any one (or more) of the faulty machines (or the good machine) assumes a different value on line a. For multi-valued logic, parallel fault simulation requires a more complex data structure using several computer words to represent a network signal.

Deductive fault simulation [Armstrong 1972, Chappell 1976], operates according to the principle that when the input values to a logic element as well as the effects of network faults on those input values are known, the output values of the element under fault conditions can be deduced. It uses the concept of fault list propagation. In two-valued deductive simulation the good machine is simulated explicitly and a list of faults L_a is associated with each network signal a. L_a lists all faults which produce an error on signal a. If enough storage is available, all faults can be processed in one simulation pass. The simulator is table-driven and event-directed. When three values (or more) are employed, the complexity of deductive simulation greatly increases.

Concurrent fault simulation, which was proposed by Ulrich and Baker ([Ulrich 1973, Ulrich 1974]), processes both the good machine and the faulty machines concurrently. It is possible to simulate all faults in one simulation pass. A super fault list S_a is associated with signal line a. Each entry in S_a (corresponding to a faulty machine) for an output line a of an e.ement E, contains the input values and the output value of E for that faulty machine. An entry is formed in S_a if and only if the fault causes one or more of the input signals or the output signal a of E to have a different value from that of the good machine. Fault lists are longer than for deductive simulation, therefore more storage is required. The entries in S_a are simulated one at a time (separately) as long as there is activity associated with them. Therefore the concurrent simulator only processes the active networks. The same evaluation routine is used to process the good element as well as the element affected by a fault. A concurrent simulator can therefore employ the same techniques developed for processing multi-valued logic, delays, mixed-logic simulation, etc. as used in a true value simulator.

CHAPTER 5 COMPARISON BETWEEN FAULT SIMULATION TECHNIQUES

The single-fault simulation technique is very time consuming and therefore it is in very little use today. Its main advantage is that an available logic simulator for good machines can be used to process faulty machines, i.e., there is no need to develop a fault simulator.

The following is a comparison between the parallel, deductive and concurrent fault simulators. They have all been used in industry, with the concurrent simulator gaining in popularity over the other two techniques ([Chappell 1974, El-Zig 1979, Miura 1978, Abramovici 1977, Shuler 1977, Shuler 1979, d'Abreu 1980, Ulrich 1980, Giambasi 1980]).

In terms of execution speed, concurrent fault simulation appears to be superior to the other techniques for either two-valued logic or multi-valued logic. For two-valued logic, parallel fault simulation appears to be faster than deductive simulation when simulating small (e.g., less than 500 gates) highly sequential networks; otherwise deductive simulation is faster [Chappell 1974]. For three-valued logic it appears that deductive simulation is no faster than parallel. Some major reasons for the differences in speed are: (a) parallel simulation requires more simulation passes than either deductive or concurrent; (b) processing of fault lists is much simpler for concurrent than deductive simulation; (c) since each entry in S_a of a concurrent fault list is processed separately and its inputs are known, fast simulation techniques such as table look-up can be used; (d) concurrent simulation processes only active signals and elements. Assume, for example, an active element in some faulty circuit. A parallel simulator would reevaluate that element for the good machine and all the other faulty machines represented by a computer word, even though these values do not change. In deductive simulation the entire fault list for that element has to be processed even though only one fault produces activity. In the concurrent simulation, only the single active fault is processed.

In terms of memory requirements concurrent simulation requires the most memory and parallel the least memory. Memory requirements in deductive and concurrent simulators are dictated by the lengths of the fault lists. The fault lists in a concurrent simulator contain more information, and are therefore more complex, than the fault lists in a deductive simulator.

In terms of compatibility with functional models, concurrent simulation is by far superior to parallel and deductive, while deductive is the least suitable. In parallel simulation, since independence between the bits of a word must be preserved, evaluation of functional elements that require non-Boolean operations cannot be executed in parallel. This implies that for many functional elements faults must be propagated through the element one at a time, defying the purpose of parallel simulation. A similar problem exists for the deductive simulator. The functional deductive fault simulator reported by Chappell et al. [Chappell 1976] models a functional element at the gate level in order to propagate faults through it, thus defeating the purpose of functional modeling. A concurrent simulator is easily adaptable to functional simulation since each element in a fault list is handled individually.

In terms of compatibility with multi-valued logic, concurrent simulation is most suitable while deductive is least suitable.

In terms of timing accuracy, concurrent simulation is superior to both parallel and deductive. In parallel simulation the modeling of timing faults is very difficult and the precise modeling of different rise and fall times for faults is not possible. In deductive simulation time dependent faults cannot be processed individually as is possible in concurrent simulation.

Fault simulation methods are compared in [Levendel 1979], in terms of their accuracy in the presence of unknown signals. The deductive method is shown to be less accurate than the other methods, which are shown to be equivalent.

Finally, concurrent simulation is superior to both parallel and deductive simulation in terms of adaptability of the simulator to new fault models, new logic primitives or new logic values. Concurrent fault simulation appears to be the most promising technique for future simulators. However, great care is required in element evaluation, scheduling and in the processing of fault lists in order to obtain an efficient concurrent fault simulator.

CHAPTER 6 CURRENT USE OF FAULT SIMULATION

A study by Breuer et al. [Breuer 1981] of the state-of-the-art of design automation in 1978-1979 surveyed 15 companies and government laboratories, mostly in the U.S., with a few in Japan and Europe. The survey found that for fault simulators used for PC boards and LSI chips most systems employ parallel fault simulation (11 systems). Two use concurrent and one is under development. The only deductive simulator is that which is part of the D-LASAR system. All simulators process stuck-at-faults even though LSI circuits require more complex fault models. Most simulators process only 3 logic values (0, 1, u (unknown)) with only one simulator processing 2 logic values and five processing 4 logic values (0,1,u,Z (high impedance)). In terms of delay modeling only three simulators are restricted to unit delay, with five allowing for assignable delays and seven allowing for assignable rise/fall delays. These systems process networks which are considerably smaller than those used for fault-free simulation. Most simulators can handle networks of up to 5K gates, with the D-LASAR deductive simulator capable of handling up to 75K gates, and with another deductive simulator and a concurrent simulator capable each of handling 10K gates.

In summary, the present status of fault simulation can be summarized as follows:

1. Large networks can be simulated fairly efficiently.
2. It is difficult to develop high level primitive models.
3. The fault modes used are restricted.

4. It is difficult to handle mixed levels of descriptions.

5. Timing is not always accurately processed, particularly in terms of races, hazards, spike rejection, etc.

6. Modeling of pass transistors and bi-directional pins is difficult.

CHAPTER 7 CONCLUSIONS

VLSI is introducing great difficulties in the area of fault simulation. Some of the major reasons for these difficulties are as follows:

1. The great increase in circuit complexity makes the simulation of all gate-level faults very time consuming.
2. Consideration of only single stuck-at faults may be inadequate. Multiple faults, non stuck-type faults and intermittent faults are more important. Such fault modes are either difficult or impractical to process.
3. The gate-level description of LSI/VLSI chip may not be available to a test engineer.

Though the future of fault simulation seems uncertain, there is as yet no good substitute. Some major trends in fault simulation and requirements for simulation of VLSI circuits are:

1. Allow for higher levels and mixed levels of modeling.
2. Allow the processing of more complex fault modes.
3. Allow for dynamic switching of models.
4. Develop new techniques to improve simulation efficiency and accuracy.
5. Use concurrent fault simulation.

CHAPTER 8 REFERENCES

1. [Abramovici 1977] M. Abramovici et al., "Concurrent Fault Simulation and Functional Level Modeling," Proc. 14th Design Automation Conference, pp. 128-137, June 1977.

2. [Armstrong 1972] D.B. Armstrong, "A Deductive Method for Simulating Faults in Logic Circuits," IEEE Trans. Computers, vol. C-21, pp. 464-471, May 1972.

3. [Breuer 1972] M.A. Breuer, "A Note on Three Valued Logic Simulation," IEEE Trans. Computers, vol. C-21, pp. 399-402, April 1972.

4. [Breuer 1976] M.A. Breuer and A.D. Friedman, Diagnosis and Reliable Design of Digital System, Chapter 4, Computer Science Press, 1976.

5. [Breuer 1981] M.A. Breuer et al., "A Survey of the State-of-the-Art of Design Automation of Digital Systems," Computer, October 1981.

6. [Chappell 1974] S.G. Chappell et al., "Comparison of Parallel and Deductive Simulation Techniques," IEEE Trans. Computers, vol. C-23, pp. 1131-1139, November 1974.

7. [Chappell 1976] S.G. Chappell et al., "Functional Simulation in the LAMP System," Proc. 13th Design Automation Conference, pp. 42-47, June 1976.

8. [d'Abreu 1980] M.A. d'Abreu and E.W. Thompson, "An Accurate Functional Level Concurrent Fault Simulator," Proc. 17th Design Automation Conference, pp. 210-217, June 1980.

9. [Eichelberger 1965] E.B. Eichelberger, "Hazard Detection in Combinational and Sequential Switching Circuits," IBM J. Res. Develop., vol. 9, pp. 90-99, March 1965.

10. [El-Ziq 1979] Y.M. El-Ziq, "Testing of MOS Combinational Networks. A Procedure for Efficient Fault Simulation and Test Generation," Proc. 16th Design Automation Conference, pp. 162-170, June 1979.

11. [Friedes 1970] A. Friedes, "The Propagation of Fault Lists through Combinational or Sequential Circuits," Proc. Workshop on Fault Detection and Diagnosis in Digital Circuits and Systems, Lehigh University, pp. 12-41, December 7-9, 1970.

12. [Giambiasi 1980] N. Giambiasi et al., "Methods for Generalized Deductive Fault Simulation," Proc. 17th Design Automation Conference, pp. 386-393, June 1980.

13. [Hong 1979] S.J. Hong, "Fault Simulation Strategy for Combinational Logic Networks," Proc. 16th Design Automation Conference, pp. 96-102, June 1979.

14. [Levendel 1979] Y.H. Levendel and P.R. Menon, "Unknown Signal Values in Fault Simulation, " Proc. 16th Design Automation Conference, pp. 125-128, June 1979.

15. [Malek 1978] M. Malek and A.K. Bose, "Functional Simulation and Fault Diagnosis," Proc. 15th Design Automation Conference, pp. 340-346, June 1978.

16. [Miara 1978] A. Miara and N. Giambiasi, "Dynamic and Deductive Fault Simulation," Proc. 15th Design Automation Conference, pp. 439-443, June 1978.

17. [Mourad 1980] J.S. Mourad, "An Optimized ATPG," Proc. 17th Design Automation Conference, pp. 381-385, June 1980.

18. [Northcutt 1980] J.D. Northcutt, "The Design and Implementation of Fault Insertion Capabilities for ISPS," Proc. 17th Design Automation Conference, pp. 197-209, June 1980.

19. [Schuler 1977] D.M. Schuler and R.K. Cleghorn, "An Efficient Method of Fault Simulation for Digital Circuits Modeled from Boolean Gates and Memories," Proc. 14th Design Automation Conference, pp. 230-238, June 1977.

20. [Schuler 1979] D.M. Schuler et al., "A Program for the Simulation and Concurrent Fault Simulation of Digital Circuits Described with Gates and Functional Models," Proc. 1979 Test Conference, pp. 203-207, October 1979.

21. [Seshu 1962] S. Seshu and D.N. Freeman, "The Diagnosis of Asynchronous Sequential Switching Systems," IEEE Trans. Computers, vol. EC-11, pp. 459-465, August 1962.

22. [Strange 1977] J.J. Strange, "Fault Modeling in a Hierarchical Simulator," Proc. 14th Design Automation Conference, pp. 118-127, June 1977.

23. [Szygenda 1970] S.A. Szygenda et al., "A Model and Implementation of a Universal Time Delay Simulator for Large Digital Nets," Proc. SJCC, pp. 207-216, 1970.

24. [Szygenda 1972a] S.A. Szygenda and E.W. Thompson, "Fault Insertion Techniques and Models for Digital Logic Simulation," Proc. FJCC, pp. 875-884, 1972.

25. [Szygenda 1972b] S.A. Szygenda, "TEGAS-2 - Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic," Proc. 9th ACM-IEEE Design Automation Workshop, Dallas, Texas, pp. 116-127, June 1972.

26. [Thompson 1975a] E.W. Thompson and S.A. Szygenda, "Three Levels of Accuracy for the Simulation of Different Fault Types in Digital Systems," Proc. 12th Design Automation Conference, pp. 105-113, June 1975.

27. [Thompson 1975b] E.W. Thompson and S.A. Szygenda, "Digital Logic Simulation, Part 2: Parallel Fault Simulation," Computer, p. 38-49, March 1975.

28. [Ulrich 1973] E.G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Networks," Proc. 10th Design Automation Conference, pp. 145-150, June 1973.

29. [Ulrich 1974] E.G. Ulrich and T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," Computer, vol. 7, pp. 39-44, April 1974.

30. [Ulrich 1980] E. Ulrich et al., "High Speed Concurrent Fault Simulation with Vectors and Scalars," Proc. 17th Design Automation Conference, pp. 374-380, June 1980.